gemalto
security to be free

HSMoD Service

ORACLE DATABASE INTEGRATION GUIDE

Document Information

Product Version	1.7
Document Part Number	007-013897-001
Release Date	21 November 2018

Revision History

Revision	Date	Reason
Rev. A	17 August 2017	For initial release 1.1.0
Rev. B	19 September 2017	For release 1.1.1
Rev. C	14 November 2017	For release 1.2
Rev. D	05 February 2018	For release 1.3
Rev. E	02 March 2018	For HSM on Demand release 1.3
Rev. F	05 April 2018	For release 1.4
Rev. G	07 May 2018	For HSM on Demand release 1.4
Rev. H	10 June 2018	For release 1.5
	12 September 2018	For release 1.6
	21 November 2018	For release 1.7

The information contained in this document is provided “AS IS” without any warranty of any kind. Unless otherwise expressly agreed in writing, Gemalto makes no warranty as to the value or accuracy of information contained herein.

The document could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Furthermore, Gemalto reserves the right to make any change or improvement in the specifications data, information, and the like described herein, at any time.

Gemalto hereby disclaims all warranties and conditions with regard to the information contained herein, including all implied warranties of merchantability, fitness for a particular purpose, title and non-infringement. In no event shall Gemalto be liable, whether in contract, tort or otherwise, for any indirect, special or consequential damages or any damages whatsoever including but not limited to damages resulting from loss of use, data, profits, revenues, or customers, arising out of or in connection with the use or performance of information contained in this document.

Gemalto does not and shall not warrant that this product will be resistant to all possible attacks and shall not incur, and disclaims, any liability in this respect. Even if each product is compliant with current security standards in force on the date of their design, security mechanisms' resistance necessarily evolves according to the state of the art in security and notably under the emergence of new attacks. Under no circumstances, shall Gemalto be held liable for any third party actions and in particular in case of any successful attack against systems or equipment incorporating Gemalto products. Gemalto disclaims any liability with respect to security for direct, indirect, incidental or consequential damages that result from any use of its products. It is further stressed that independent testing and verification by the person using the product is particularly encouraged, especially in any application in which defective, incorrect or insecure functioning could result in damage to persons or property, denial of service, or loss of privacy.

SafeNet Data Protection on Demand1.7

ORACLE DATABASE INTEGRATION GUIDE

Contents

Overview

This document provides detailed procedures on integrating your Oracle Database with an HSM on Demand Service and enabling transparent data encryption (TDE). It provides the necessary information to install, configure, and integrate the Oracle Database with an HSM on Demand Service.

This document contains the following sections:

This overview contains the following sections:

Third Party Application Details

This integration guide supports the following third party applications or software:

> Oracle Database 12c R2/R1

Supported Platforms

The following platforms were tested with HSMoD Service:

Third Party Software	Operating System
Oracle Database 12C R2	Windows Server 2012 R2
	Red Hat Enterprise Linux 7 64-bit
	Solaris Sparc 11.3.64 bit
	Solaris x86 11.3.64 bit
	AIX 7.2
Oracle Database 12C R1	Windows Server 2016
	Windows Server 2012 R2
	Red Hat Enterprise Linux 7 64-bit

Preparing for the Integration

Before you proceed with the integration, ensure you have completed the following:

> "Provision your HSM on Demand Service" below
> "Enable High-Availability (High-availability configurations only)" on page 10
> "Set up Oracle Database " on page 11
> "Set up HSM on Demand Service for Transparent Data Encryption" on page 11
> "Export Oracle Variables" on page 11
> "Update Oracle HSM Heartbeat Timeout" on page 11

Provision your HSM on Demand Service

The HSM on Demand Service provides your client machine with access to an HSM application partition for storing cryptographic objects used by your applications. Application partitions can be assigned to a single client, or multiple clients can be assigned to, and share, a single application partition.

You must provision your HSM on Demand service by adding the service, downloading the service client package and initializing the HSM. Provisioning your HSM on Demand service entails:

> "Adding a Service" below
> "Adding a Service Client" on the next page
> "Initializing the HSM" on page 9

Adding a Service

1. Under the **Services** tab, select the **Add New Service** page. Click **Deploy** on the service tile for the service you wish to add.

   NOTE Click **Deploy** on the HSM on Demand Service tile for your integration.

2. Review the "Terms of Services DPoD." Enable the **I have read and accept the Terms of Service above** check box and then click **Next**.
3. On the **Add <service_type> Service** page, enter a name for the Service in the **Service Name** field. You can optionally allow non-FIPS approved algorithms by selecting the **Allow non-FIPS approved algorithms** check box. Click **Next**.

   CAUTION! You cannot alter the FIPS setting after creating the service. You must decide if the service should allow or disallow non-FIPS approved algorithms before clicking **Finish** in the next step.

4. Review the configuration summary page. If acceptable, click **Finish**. If you would like to make changes to the configuration, click **Go Back**.

When completed, the new service is listed under **My Services** and a **Create Service Client?** window displays.

5. Click **Create Service Client**.

Adding a Service Client

1. In the **Create Service Client** window enter a name for the service client in the **Service Client Name** field.

> **NOTE** If the **Create Service Client** window is not available, navigate to the **Services** tab and click the name of the Service you would like to generate a client for in the **My Services** table. On the Service Details page, click **New Service Client**.

2. Select **Create Service Client**.

   A new HSM service client package is created and provided for downloading on your client system.

> **NOTE** The HSM service client package is a zip file that contains system information needed to connect your client system to an existing HSM on Demand service. The HSM service client package should download immediately on creation. If it does not, or you lose access to your HSM service client package it can be accessed or reacquired through the **My Services** table.

3. Transfer the service client package to your client system. You can use SCP, PSCP, WinSCP, FTPS, or any other secure file transfer tool.
4. Unzip the service client package.

   For Linux, enter:

   ```
   unzip <service_client_package>.zip
   ```

   For Windows, using the Windows GUI or an unzip tool unzip the file:

   ```
   <service_client_package>.zip
   ```

> **NOTE** For more information about the service client package contents see .

5. Extract the cvclient-min file.

> **NOTE** Extract the cvclient-min file in the directory where you extracted the <service_client_package>.zip. **Do not** extract to a new cvclient-min directory.

   For Linux, untar the cvclient-min.tar

   ```
   tar xvf cvclient-min.tar
   ```

   For Windows, unzip the cvclient-min.zip.

6. Set the environment variable.

   For Linux, execute:

```
source ./setenv
```

For Windows, right click setenv.cmd and select **Run as Administrator**.

> **NOTE** If you encounter the error dll load failed with GetLastError() 126 move the contents of the cvclient_min folder up one directory and execute setenv.

7. Start LunaCM.

   For Linux, execute the following from the directory where you extracted the cvclient-min.tar file.

   ```
   ./bin/64/lunacm
   ```

   For Windows, execute the following from the directory where you unzipped the cvclient-min.zip file.

   ```
   lunacm
   ```

Initializing the HSM

1. Set the active slot to the service partition.

   ```
   lunacm:>slot set -slot <slot_number>
   ```

   > **NOTE** Execute slot list in LunaCM to identify the slot number associated with your service.

2. Initialize the application partition. During this process you will create the partition's Security Officer (SO), set the SO password, and specify the cloning domain.

   ```
   lunacm:> partition init -label <service_label>
   ```

3. Optional: If you wish to transfer key material to or from a PED-authenticated Luna partition, you initialize the SafeNet Data Protection On Demand partition using the red PED domain key.

   a. For DPoD deployments, contact customer support to obtain the necessary PED drivers so that your HSM client can communicate with the PED.

   b. Attach the PED locally to the client computer, insert the red cloning domain PED key, and initialize the partition, including the option to set the cloning domain from the red PED key. Execute:

      ```
      lunacm:> partition init -label <cryptovisor_partition_label> -importpeddomain
      ```

4. Log in as the partition's Security Officer:

   ```
   lunacm:>role login -name Partition SO
   ```

5. Initialize the Crypto Officer role:

   ```
   lunacm:>role init -name Crypto Officer
   ```

6. Log out of the partition Security Officer role and log in as the Crypto Officer.

   ```
   lunacm:>role logout
   lunacm:>role login -name Crypto Officer
   ```

7. You must change the Crypto Officer password immediately on the initial log in. Failure to due so will result in a password error on subsequent logins.

   ```
   lunacm:>role changepw -name Crypto Officer
   ```

8. Initialize the Crypto User role:

```
lunacm:>role init -name Crypto User
```

9. Log out of the partition Crypto Officer role and log in as the Crypto User.

```
lunacm:>role logout
lunacm:>role login -name Crypto User
```

10. You must change the Crypto User password immediately on the initial log in. Failure to do so will result in a password error on subsequent logins.

```
lunacm:>role changepw -name Crypto User
```

This completes initializing the HSM on Demand Service. The Crypto Officer and Crypto User roles can now be used to integrate applications with the HSMoD service to perform cryptographic operations

Constraints on HSMoD Services

Please take the following limitations into consideration when integrating your application software with an HSM on Demand Service.

HSM on Demand Service in FIPS mode

HSMoD services operate in a FIPS and non-FIPS mode. If your organization requires non-FIPS algorithms for your operations, ensure you enable the **Allow non-FIPS approved algorithms** check box when configuring your HSM on Demand service. The FIPS mode is enabled by default.

Refer to the *Mechanism List* in the SDK Reference Guide for more information about available FIPS and non-FIPS algorithms.

Verify HSM on Demand <slot> value

LunaCM commands work on the current slot. If there is only one slot, then it is always the current slot. If you are completing an integration using HSMoD services, you need to verify which slot on the HSMoD service you send commands to. If there is more than one slot, then use the **slot set** command to direct a command to a specified slot. You can use **slot list** to determine which slot numbers are in use by which HSMoD service.

Enable High-Availability (High-availability configurations only)

You must enable the HAOnly setting for failover to function properly. If you are using an HSM on Demand Service, you must enable the following setting in the Crystoki.conf (UNIX) or Crystoki-template.ini (Windows) file.

UNIX:

```
Misc = {
PE1746Enabled=0;
}
```

Windows:

```
[Misc]
PE1746Enabled=0
```

Set up Oracle Database

The Oracle Database must be installed on the target machine to complete the integration process. Refer to the *Oracle Database Documentation* for detailed installation procedures.

Set up HSM on Demand Service for Transparent Data Encryption

You must copy the HSMoD Service PKCS##11 library to the specified directory structure. We recommend using the following directory structure:

UNIX	**/opt/oracle/extapi/**<32,64>**/hsm/**<vendor>**/**<version>**/lib**<PKCS#11lib>**.ext**
Windows	**%SYSTEMDRIVE%\oracle\extapi**<32,64>**\hsm**<vendor>\<version>**\lib**<PKCS#11lib>**.ext**

NOTE The Oracle user must have read/write permissions at the above directory.

Export Oracle Variables

This integration guide uses the following Oracle environment variables. We recommend that you export the same environment variables for the purpose of following the integration guide.

```
export ORACLE_SID=orcl
export ORACLE_BASE=/u01/app/oracle (oracle installation directory)
export ORACLE_HOME=$ORACLE_BASE/product/12.1.0/dbhome_1
export PATH=$PATH:$ORACLE_HOME/bin
export TNS_ADMIN=$ORACLE_HOME/network/admin
```

Update Oracle HSM Heartbeat Timeout

If using a Linux Operating System, be aware of the following:

Oracle Event 28420 determines the number of HSM heartbeats that can fail before the Oracle wallet is closed. Because the HSM heartbeat fires every 3 seconds, a very short network outage may result in wallet closure. As a result, we recommend increasing the number of possible HSM heartbeat failures before the wallet closes. To make this a default setting execute:

```
ALTER SYSTEM SET EVENT='28420 trace name context <quantity> level <timeout_count>' COMMENT='HSM
heartbeat timeout attempt' SCOPE=SPFILE;
```

NOTE You can change the <timeout_count> from 1 to any value of your choice. If the <timeout_count> is set to 10, then the RDBMS will allow 10 heartbeats to fail before closing the wallet..

Integrating Oracle Transparent Data Encryption (TDE) Database with an HSM on Demand Service

The HSM on Demand Service is used to secure the Master Encryption Key for Oracle Transparent Data Encryption (TDE) in a FIPS 140-2 approved HSM.

TDE allows you to encrypt sensitive data in database table columns or application tablespaces. We recommend securing the columns on the Oracle database with TDE using an HSM on Demand Service for the following reasons:

> It secures the master encryption key so that it is never exposed in insecure memory

> The HSM on Demand Service is a more secure alternative to the Oracle wallet.

There are two methods of approach:

> "Migrating the Master Encryption Key for HSM-based Encryption " below

> "Generating the Master Encryption Key for HSM-based Encryption" on page 15

After you acquire the key, you must complete the following:

> "Setting up Oracle to create Auto-Open HSM" on page 17

Migrating the Master Encryption Key for HSM-based Encryption

To use HSM-based encryption you require a master encryption key that will be stored in the HSM on Demand Service. The master encryption key is used to encrypt or decrypt the column or tablespace. To migrate an existing master encryption key for HSM-based encryption you must:

> "Configuring the software keystore " below

> "Verifying that the database can reach the HSM on Demand Service" on the next page

> "Creating the HSM auto wallet " on page 14

> **NOTE** This procedural set assumes that no software based wallet or HSM based wallet exists.

Configuring the software keystore

You need to configure a software keystore to store the encryption keys for your Oracle database.

To configure the software keystore wallet

1. Open the **sqlnet.ora** file in a text editor and add the following:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA = (DIRECTORY = <path_to_
oracle_wallet_directory>)))
```

2. Log into the database as the administrator or operator.

```
$sqlplus / as sysdba
```

3. Grant the ADMINISTER KEY MANAGEMENT privilege to SYSTEM and any additional users that you wish to configure access.

```
SQL> GRANT ADMINISTER KEY MANAGEMENT TO SYSTEM;
SQL> commit;
```

4. Connect to the database as system.

```
SQL> connect system/<password>
```

> **NOTE** The password for the system user is set during the Oracle database installation.

5. Create the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE <keystore_location> IDENTIFIED BY <software_
keystore_password>;
```

> **NOTE** <keystore_location is the path to the Oracle wallet directory that is defined in the **sqlnet.ora** file. The <software_keystore_password> requires at least 8 characters.

6. Open the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY <software_keystore_password>;
```

7. Set the master encryption key in the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY <software_keystore_password> WITH BACKUP
USING <backup_domain>;
```

8. Exit SQL.

```
SQL> exit
```

Verifying that the database can reach the HSM on Demand Service

You need to verify that your Oracle configuration can communicate with the HSM on Demand Service.

To verify that the database can reach the HSM on Demand Service

1. Open the **sqlnet.ora** file in a text editor and edit the following:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY = <path_to_
oracle_wallet_directory>)))
```

2. Start the database
3. Connect to the database as system.

```
SQL> connect system/<password>
```

4. Migrate the keystore onto the HSM device.

```
SQL> ADMINISTER KEY MANAGEMENT SET ENCRYPTION KEY IDENTIFIED BY <HSM_partition_password>
MIGRATE USING <software_keystore_password> WITH BACKUP USING <backup_domain>;
```

5. Alter the <software_keystore_password> so it is the same as the <HSM_partition_password>.

```
SQL> ADMINISTER KEY MANAGEMENT ALTER KEYSTORE PASSWORD IDENTIFIED BY <software_keystore_
password> SET <HSM_partition_password> WITH BACKUP USING <backup_domain>;
```

NOTE Now, when you open the software keystore, it will open both the software-based keystore, as well as the HSM-based keystore

6. Close the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY <HSM_partition_password>;
```

7. Open the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY <HSM_partition_password>;
```

8. Create an auto-login keystore for the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE <keystore_location>
IDENTIFIED BY <HSM_partition_password>;
```

NOTE To restrict the auto-login wallet to the local system, substitute LOCAL AUTO_LOGIN for AUTO_LOGIN

9. Verify that an auto-open software keystore has been created in the oracle wallet directory specified in the **sqlnet.ora** file. You will find two wallets in this directory. Move or rename the encryption wallet to ensure that Oracle uses the auto-login wallet.
10. Restart the database and log in as system. Open the HSM keystore and the software wallet will open automatically.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY <HSM_partition_password>;
```

Creating the HSM auto wallet

You can configure your Oracle database to open the wallet on the HSM on Demand Service.

To create the HSM auto wallet

1. Open the **sqlnet.ora** file in a text editor and update the following:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA = (DIRECTORY =
/etc/oracle/wallet)))
```

2. Rename or move the Auto-open wallet from the location mentioned in the **sqlnet.ora** file. Move the encryption wallet into the wallet directory.
3. Restart the database and connect as system.
4. Open the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY <HSM_partition_password>;
```

5. Add the HSM secret to the client.

```
SQL> ADMINISTER KEY MANAGEMENT ADD SECRET <HSM_partition_password> FOR CLIENT 'HSM_PASSWORD'
IDENTIFIED BY <software_keystore_password> WITH BACKUP USING <backup_domain>;
```

6. Close the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY <software_keystore_password>;
```

7. Create (or recreate) the Auto-login keystore.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE <keystore_location>
IDENTIFIED BY <software_keystore_password>
```

8. Open the **sqlnet.ora** file in a text editor and update the following:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY =
/etc/oracle/wallet)))
```

9. Restart the database and connect as system.

```
SQL> connect system/<password>
```

10. Verify the wallet information.

```
SQL> SELECT * FROM V$ENCRYPTION_WALLET;
```

Generating the Master Encryption Key for HSM-based Encryption

To use HSM-based encryption you require a master encryption key that will be stored in the HSM on Demand Service. The master encryption key is used to encrypt or decrypt the column or tablespace. To generate a master encryption key for HSM-based encryption you must:

> "Configuring Oracle to generate the master encryption key on the HSM on Demand Service" below
> "Creating the HSM auto wallet " on the previous page

NOTE This procedural set assumes that no software based wallet or HSM based wallet exists.

Configuring Oracle to generate the master encryption key on the HSM on Demand Service

You can configure Oracle to generate the master encryption key from the HSM on Demand Service. If you do not configure Oracle to use the HSM on Demand Service it will use its own encryption key.

To configure Oracle to generate the master encryption key on the HSM on Demand Service

1. Open the **sqlnet.ora** file in a text editor and add the following:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM))
```

2. Log into the database as the administrator .

```
$ sqlplus / as sysdba
```

3. Grant the ADMINISTER KEY MANAGEMENT privilege to SYSTEM and any additional users that you wish to configure access.

```
SQL> GRANT ADMINISTER KEY MANAGEMENT TO SYSTEM;
SQL> commit;
```

4. Connect to the database as system.

```
SQL> connect system/<password>
```

NOTE The password for the system user is set during the Oracle database installation.

5. Open the HSM keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY <HSM_partition_password>;
```

> **NOTE** To select from multiple [[[Undefined variable Default.slot]]], use the syntax <HSM_partition_password>|<partition_name>

6. Set the master encryption key in the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY <HSM_partition_password>;
```

7. Open the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY <HSM_partition_password>;
```

8. Exit SQL.

```
SQL> exit
```

Creating the HSM auto wallet

You can configure your Oracle database to automatically open the wallet on the HSM on Demand Service and access the encryption key.

To create the HSM auto wallet

1. Close the HSM keystore, if it is open.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY <HSM_partition_password>;
```

2. Open the **sqlnet.ora** file in a text editor and update the following:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA = (DIRECTORY =
/etc/oracle/wallet)))
```

3. Create a software keystore in the location defined in the **sqlnet.ora** file.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/etc/oracle/wallet' IDENTIFIED BY <software_
keystore_password>;
```

4. Open the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY <software_keystore_password>;
```

5. Add the HSM secret as a client.

```
SQL> ADMINISTER KEY MANAGEMENT ADD SECRET <HSM_partition_password> FOR CLIENT 'HSM_PASSWORD'
IDENTIFIED BY <software_keystore_password> WITH BACKUP USING <backup_domain>;
```

6. Close the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY <software_keystore_password>;
```

7. Create auto-login keystore

```
SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE '/etc/oracle/wallet'
IDENTIFIED BY <software_keystore_password>;
```

8. Open the **sqlnet.ora** file in a text editor and update the following:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY =
/etc/oracle/wallet)))
```

NOTE After restarting the database, the next time a TDE operation executes, the HSM auto-login keystore will open automatically.

9. Restart the database and connect as system.

```
SQL> connect system/<password>
```

10. Verify the wallet information.

```
SQL> SELECT * FROM V$ENCRYPTION_WALLET;
```

Setting up Oracle to create Auto-Open HSM

The following sectional use cases cover example Oracle Database configurations and their status. It provides procedural material for integrating the database with an HSM on Demand Service based on the Oracle Database's current status. It provides two sets of commands for the user where syntax varies, Standard and RAC. The Standard line provides syntax for completing an integration with the Oracle standalone database. The RAC line provides syntax for configuring on an Oracle RAC database.

Scenario 1. TDE is used in HSM only mode (never migrated from an Oracle Wallet)

1. Open the **sqlnet.ora** file in a text editor and update the ENCRYPTION_WALLET_LOCATION section to be the following:

Standard	`ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY = /etc/ORACLE/WALLETS/$ORACLE_SID)))`
RAC	`ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY = /etc/oracle/wallet/RAC)))`

2. Create a local auto-open and encryption wallet in the directory defined in the **sqlnet.ora** file.

Standard	`/etc/ORACLE/WALLET/$ORACLE_SID #orapki wallet create -wallet . -auto_login`
RAC	`/etc/oracle/wallet/RAC))) #orapki wallet create -wallet . -auto_login`

3. Add the following entry to the wallets to enable auto-opening of the HSM.

```
# mkstore -wrl . -createEntry ORACLE.TDE.HSM.AUTOLOGIN <any_non_empty_string>
```

4. Move the encryption wallet out of the ENCRYPTION_WALLET_LOCATION defined in the **sqlnet.ora** file.
5. Close the connection to the HSM.

```
SQL> alter system set encryption wallet close identified by <HSM_partition_password>
```

6. Open the connection to the HSM.

```
SQL> alter system set encryption wallet open identified by <HSM_partition_password>
```

NOTE This command inserts the <HSM_partition_password> into the auto-open wallet. As a result, if the wallet is open, no password is required to access data encrypted with the TDE master encryption key.

Scenario 2. TDE is used for the first time

1. Open the **sqlnet.ora** file in a text editor and add the following

Standard	`ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY = /etc/ORACLE/WALLETS/$ORACLE_SID)))`
RAC	`ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY = /etc/oracle/wallet/RAC)))`

2. Create a local auto-open and encryption wallet in the directory defined in the **sqlnet.ora** file.

Standard	`/etc/ORACLE/WALLET/$ORACLE_SID #orapki wallet create -wallet . -auto_login`
RAC	`/etc/oracle/wallet/RAC))) #orapki wallet create -wallet . -auto_login`

3. Add the following entry to the wallets to enable auto-opening of the HSM.

```
# mkstore –wrl . –createEntry ORACLE.TDE.HSM.AUTOLOGIN <any_non_empty_string>
```

4. Move the encryption wallet out of the ENCRYPTION_WALLET_LOCATION defined in the **sqlnet.ora** file.
5. Generate a TDE master encryption key inside the HSM.

```
SQL> alter system set encryption key identified by <HSM_partition_password>
```

Scenario 3. HSM and encryption wallet already in use

1. Ensure that the master encryption key has been migrated to the HSM device and the **sqlnet.ora** file contains the following entry:

Standard	`ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY = /etc/ORACLE/WALLETS/$ORACLE_SID)))`
RAC	`ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY = /etc/oracle/wallet/RAC)))`

2. From the encryption wallet directory, create an auto-open wallet.

```
# orapki wallet create – wallet . –auto_login
```

3. Execute the following command to enable the auto-open of the HSM wallet.

```
# mkstore –wrl . –createEntry ORACLE.TDE.HSM.AUTOLOGIN <any_non_empty_string>
```

4. Move the encryption wallet out of the directory defined in the **sqlnet.ora** file.
5. Close the connection to the HSM.

```
SQL> alter system set encryption wallet close identified by "<HSM_partition_password>";
```

6. Open the connection to the HSM.

```
SQL> alter system set encryption wallet open identified by "<HSM_partition_password>";
```

NOTE This command inserts the <HSM_partition_password> into the auto-open wallet. As a result, if the wallet is open, no password is required to access data encrypted with the TDE master encryption key.

Scenario 4. HSM and (local) auto-login wallet already in use

1. Ensure that the master encryption key has been migrated to the HSM device, a (local) auto-login wallet exists, the encryption wallet has been moved from the ENCRYPTION_WALLET_LOCATION, and the **sqlnet.ora** file contains the following entry:

Standard	`ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY = /etc/ORACLE/WALLETS/$ORACLE_SID)))`
RAC	`ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY = /etc/oracle/wallet/RAC)))`

2. Restore the encryption wallet from its secure location. Return the encryption wallet to the location defined in the **sqlnet.ora** file.
3. Execute the following command to enable the auto-open of the HSM wallet.

```
# mkstore -wrl . -createEntry ORACLE.TDE.HSM.AUTOLOGIN <any_non_empty_string>
```

4. Move the encryption wallet out of the directory defined in the **sqlnet.ora** file.
5. Close the connection to the HSM.

```
SQL> alter system set encryption wallet close identified by "<HSM_partition_password>";
```

6. Open the connection to the HSM.

```
SQL> alter system set encryption wallet open identified by "<HSM_partition_password>";
```

> **NOTE** This command inserts the <HSM_partition_password> into the auto-open wallet. As a result, if the wallet is open, no password is required to access data encrypted with the TDE master encryption key.

Scenario 5. Encryption wallet in use; no HSM

1. Ensure the **sqlnet.ora** file contains the following entry.

Standard	`ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY = /etc/ORACLE/WALLETS/$ORACLE_SID)))`
RAC	`ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY = /etc/oracle/wallet/RAC)))`

2. From the encryption wallet directory, create an auto-open wallet.

```
# orapki wallet create – wallet . –auto_login
```

Scenario 6. (local) Auto-login wallet in use

1. Execute the following to enable the auto-open of the HSM wallet.

```
# mkstore -wrl . -createEntry ORACLE.TDE.HSM.AUTOLOGIN <and_non_empty_string>
```

2. Ensure the **sqlnet.ora** file contains the following entry.

Standard	`ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY = /etc/ORACLE/WALLETS/$ORACLE_SID)))`

RAC	`ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY = /etc/oracle/wallet/RAC)))`

3. Migrate the TDE column master encryption key from the wallet to the HSM.

   ```
   SQL> alter system set encryption key identified by "<HSM_partition_password>" migrate using
   "<wallet_password>";
   ```

4. Move the encryption wallet out of the ENCRYPTION_WALLET_LOCATION defined in **sqlnet.ora**. If you are configuring an Oracle RAC instance, copy both the encryption wallet and the auto-login wallet to the same directory on the remaining RAC instances.

Integrating Oracle RAC with an HSM on Demand Service

Oracle Real Application Clusters (Oracle RAC) enables an Oracle database to run across a cluster of servers, providing fault tolerance, performance, and scalability with no application changes necessary. Oracle RAC provides high availability for applications by removing the single point of failure with a single server.

Oracle Clusterware is installed into a single home directory, which is called the Grid home. Oracle Clusterware enables servers, referred to as hosts or nodes, to operate as if they are one server, commonly referred to as a cluster. Although the servers are standalone servers, each server has additional processes that communicate with other servers. In this way the separate servers appear as if they are one server to applications and end users. Oracle Clusterware provides the infrastructure necessary to run Oracle RAC. The combined processing power of the multiple servers provides greater availability, throughput, and scalability than is available from a single server.

Non-cluster Oracle databases have a one-to-one relationship between the Oracle database and the instance. Oracle RAC environments, however, have a one-to-many relationship between the database and instances. Oracle RAC databases differ architecturally from non-cluster Oracle databases in that each Oracle RAC database instance also has:

- At least one additional thread of redo for each instance
- An instance-specific undo tablespace

The combined processing power of the multiple servers can provide greater throughput and Oracle RAC scalability than is available from a single server.

We recommend you familiarize yourself with the Oracle Database RAC. Read the *Oracle Database 12c RAC* product documentation for more information about installation and pre-installation requirements.

This integration contains the following topics:

Oracle Database RAC Setup

The 3 machines utilized are denoted in the setup as follows:

- RAC1.localdomain
- RAC2.localdomain
- RAC3.localdomain

This demonstration utilizes 3 Oracle homes. Each Oracle home consists of 1 database and 3 Oracle instances.

Verifying the Oracle RAC Installation

Before beginning the integration with the HSM on Demand Service we recommend verifying your Oracle RAC installation.

Verify Oracle RAC Installation

This procedural set presumes that Oracle RAC is setup and is functional.

1. You can verify that RAC is operating by executing the following commands on any Oracle RAC instance:

   /crsctl stat res –t

   srvctl config database –d RAC

2. The setup has 3 databases. They are configured as follows:

   Database ORCL1RAC

```
Database unique name: orcl1rac
Database name: orcl1rac
Oracle home: /u01/app/oracle/product/12.1.0.2/db_1
Oracle user: oracle
Spfile: +DATA/ORCL1RAC/PARAMETERFILE/spfile.301.910221979
Password file: +DATA/ORCL1RAC/PASSWORD/pwdorcl1rac.276.910221645
Domain:
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools:
Disk Groups: DATA
Mount point paths:
Services:
Type: RAC
Start concurrency:
Stop concurrency:
OSDBA group: dba
OSOPER group: oper
Database instances: orcl1rac1,orcl1rac2,orcl1rac3
Configured nodes: rac1,rac2,rac3
Database is administrator managed
```

Database ORCL2RAC

```
Database unique name: orcl2rac
Database name: orcl2rac
Oracle home: /u01/app/oracle/product/12.1.0.2/db_2
Oracle user: oracle
Spfile: +DATA/ORCL2RAC/PARAMETERFILE/spfile.331.910223671
Password file: +DATA/ORCL2RAC/PASSWORD/pwdorcl2rac.306.910223319
Domain:
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools:
Disk Groups: DATA
Mount point paths:
Services:
Type: RAC
Start concurrency:
Stop concurrency:
OSDBA group: dba
OSOPER group: oper
Database instances: orcl2rac1,orcl2rac2,orcl2rac3
Configured nodes: rac1,rac2,rac3
Database is administrator managed
```

DATABASE ORCL3RAC

```
Database unique name: orcl3rac
Database name: orcl3rac
Oracle home: /u01/app/oracle/product/12.1.0.2/db_3
Oracle user: oracle
Spfile: +DATA/ORCL3RAC/PARAMETERFILE/spfile.361.910224445
Password file: +DATA/ORCL3RAC/PASSWORD/pwdorcl3rac.336.910224083
Domain:
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools:
Disk Groups: DATA
Mount point paths:
Services:
Type: RAC
Start concurrency:
Stop concurrency:
OSDBA group: dba
OSOPER group: oper
Database instances: orcl3rac1,orcl3rac2,orcl3rac3
Configured nodes: rac1,rac2,rac3
Database is administrator managed
```

Creating the RAC directories

You must create the Oracle RAC directories to store the master encryption keys for tablespace encryption.

To create the RAC directories

1. Execute the following on each Oracle database in the RAC configuration and permit the Oracle user to access these directories on RAC1, RAC2, and RAC3.

```
# mkdir -pv /etc/oracle/wallet/ORCL1RAC
# mkdir -pv /etc/oracle/wallet/ORCL2RAC
# mkdir -pv /etc/oracle/wallet/ORCL3RAC
# cd /etc
# chown -R oracle:oinstall oracle/wallet/
# chmod -R 700 oracle/wallet/
# mkdir -pv /etc/oracle/wallet/RAC
# cd /etc
# chown -R oracle:oinstall oracle/wallet/
# chmod -R 700 oracle/wallet/
```

Acquiring and Integrating the Master Encryption Key

To complete the Oracle RAC configuration, you require access to a master encryption key. You can acquire the master encryption key in one of two methods:

> "Migrating the master encryption key onto HSM on Demand Service" below

> "Generating the master encryption key on the HSM on Demand Service" on page 27

Migrating the master encryption key onto HSM on Demand Service

To use HSM-based encryption you require a master encryption key that will be stored in the HSM. The master encryption key is used to encrypt or decrypt the column or tablespace. To migrate an existing master encryption key for HSM-based encryption you must:

> Configuring the software wallet

> Verifying the database can reach the HSM device

> Creating the HSM auto wallet

To configure the software wallet

1. On all RAC instances open the sqlnet.ora file in a text editor and add the following:

   ```
   ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA = (DIRECTORY =
   /etc/oracle/wallet/ORCL1RAC)))
   ```

2. Log into the database as the administrator.

   ```
   $ sqlplus / as sysdba
   ```

3. Grant the ADMINISTER KEY MANAGEMENT privilege to SYSTEM and any additional users that you wish to configure access.

   ```
   SQL> GRANT ADMINISTER KEY MANAGEMENT TO SYSTEM;
   SQL> commit;
   ```

4. Connect to the database as system.

   ```
   SQL> connect system/<password>
   ```

5. Create the software keystore / encryption wallet.

   ```
   SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE <keystore_location> IDENTIFIED BY <software_
   keystore_password>
   SQL> alter system set encryption key identified by "<wallet_password>";
   ```

6. Open the keystore.

   ```
   SQL> ADMINISTER KEY MANAGER SET KEYSTORE OPEN IDENTIFIED BY <keystore_password>
   ```

7. Set the master encryption key in the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY <software_keystore_password> WITH BACKUP
USING <backup_domain>;
```

8. Copy the ewallet.p12 file and backup file from RAC1 to RAC2 and RAC3 to RAC2 in the same directory as they are located in on RAC1.

```
$ scp /etc/oracle/wallet/ORCL1RAC/ewallet* oracle@rac2.localdomain:/etc/oracle/wallet/ORCL1RAC/
$ scp /etc/oracle/wallet/ORCL1RAC/ewallet* oracle@rac3.localdomain:/etc/oracle/wallet/ORCL1RAC/
```

9. Close the keystore / wallet.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY <software_keystore_password>;
SQL> ALTER SYSTEM SET ENCRYPTION WALLET CLOSE IDENTIFIED BY <wallet_password>
```

To verify the database can reach the HSM on Demand Service

1. On all RAC instances open the sqlnet.ora file in a text editor and edit the following:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY = <path_to_
oracle_wallet_directory>)))
```

2. Start the database

```
$ sqlplus / as sysdba
```

3. Connect to the database as system

```
SQL> connect system/<password>
```

4. Migrate the wallet onto the HSM device.

```
SQL> ADMINISTER KEY MANAGEMENT SET ENCRYPTION KEY IDENTIFIED BY <HSM_partition_password>
MIGRATE USING <software_keystore_password> WITH BACKUP USING <backup_domain>;
SQL> ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY <HSM_partition_password> MIGRATE USING
<wallet_password>
```

5. Copy the ewallet.p12 file and backup file from RAC1 to RAC2 and RAC3 in the same directory as they are located in on RAC1.

```
$ scp /etc/oracle/wallet/ORCL1RAC/ewallet* oracle@rac2.localdomain:/etc/oracle/wallet/ORCL1RAC/
$ scp /etc/oracle/wallet/ORCL1RAC/ewallet* oracle@rac3.localdomain:/etc/oracle/wallet/ORCL1RAC/
```

6. Alter the <software_keystore_password> so it is the same as the <HSM_partition_password>.

```
SQL> ADMINISTER KEY MANAGEMENT ALTER KEYSTORE PASSWORD IDENTIFIED BY <software_keystore_
password> SET <HSM_partition_password> WITH BACKUP USING <backup_domain>;
```

7. Copy the ewallet.p12 file and backup file from RAC1 to RAC2 and RAC3 in the same directory are they are located in on RAC1.

```
$ scp /etc/oracle/wallet/ORCL1RAC/ewallet* oracle@rac2.localdomain:/etc/oracle/wallet/ORCL1RAC/
$ scp /etc/oracle/wallet/ORCL1RAC/ewallet* oracle@rac3.localdomain:/etc/oracle/wallet/ORCL1RAC/
```

8. Close the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY <HSM_partition_password>;
```

9. Open the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY <HSM_partition_password>;
```

10. Verify the wallet information.

```
SQL> SELECT * FROM GV$ENCRYPTION_WALLET
```

11. Create the auto-login keystore for the software keystore. Create the auto-login wallet on both the RAC1 and RAC2 instances.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE <keystore_location>
IDENTIFIED BY <software_keystore_password>;
# opraki wallet create -wallet . -auto_login
```

12. Verify that the auto-login software keystore wallet was created in the oracle wallet directory specified in the sqlnet.ora file. Remove the encryption wallet from the directory so that Oracle will use the auto-login wallet.
13. Restart the database and connect as system. Open the HSM wallet and the software wallet will open automatically. keystore and the software wallet will open automatically.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY <HSM_partition_password>;
SQL> alter system set encryption wallet open identified by "HSM_partition_password";
```

To create the HSM auto wallet

1. On all RAC instances open the sqlnet.ora file in a text editor and edit the following.

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA = (DIRECTORY =
/etc/oracle/wallet/ORCL1RAC)))
```

2. Rename the cwallet.sso file on all instances. Ensure the encryption wallet is called ewallet.p12 and is available at the location defined in the sqlnet.ora file.
3. Restart the database and connect as system.
4. Open the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY <software_keystore_password>;
```

5. Add the HSM secret as a client.

```
SQL> ADMINISTER KEY MANAGEMENT ADD SECRET <HSM_partition_password> FOR CLIENT 'HSM_PASSWORD'
IDENTIFIED BY <software_keystore_password> WITH BACKUP USING <backup_identifier>;
```

> **NOTE** "HSM_PASSWORD" is an Oracle defined client name that is used to identify the HSM password as a secret in the software keystore.

6. Close the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY software_keystore_password;
```

7. Create the auto-login keystore.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE
'/etc/oracle/wallet/ORCL1RAC' IDENTIFIED BY <software_keystore_password>;
```

8. Rename the ewallet.p12 file so that Oracle uses the auto-login keystore.

> **NOTE** Copy the auto-login wallet and encryption wallet from one RAC instance to the others after creating the wallet. This is not required if you are using shared storage for the wallet.

9. On all RAC instances open the sqlnet.ora file in a text editor and update the following:

```
SQL> ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY =
/etc/oracle/wallet/ORCL1RAC)))
```

Generating the master encryption key on the HSM on Demand Service

To use HSM-based encryption you require a master encryption key that will be stored in the HSM. The master encryption key is used to encrypt or decrypt the column or tablespace. To generate a master encryption key for HSM-based encryption you must:

To generate the master encryption key on the HSM on Demand Service

1. On all RAC instances open the sqlnet.ora file in a text editor and edit the following.

   ```
   ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM))
   ```

2. Stop and start the databases.

   ```
   $ srvctl stop database -d <database_name>
   $ srvctl start database -d <database_name>
   ```

3. Verify the status of the database following the start command.

   ```
   srvctl status database -d <database_name>
   ```

4. Grant the ADMINISTER KEY MANAGEMENT privilege to SYSTEM and any additional users that you wish to configure access.

   ```
   SQL> GRANT ADMINISTER KEY MANAGEMENT TO SYSTEM;
   SQL> commit;
   ```

5. Connect to the database as system.

   ```
   SQL> connect system/<password>
   ```

6. Open the HSM keystore. Create an encryption wallet.

   ```
   SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY <HSM_partition_password>;
   ```

7. Set the master encryption key in the software keystore.

   ```
   SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY <HSM_partition_password>;
   ```

8. From an ssh connection you can run **partition contents** to verify the generated keys on the HSM device.

   partition contents

To create the HSM auto wallet

You can configure your Oracle database to automatically open the wallet on the HSM on Demand Service and access the encryption key. See for more information about configuring the HSM auto wallet. You must complete the auto wallet configuration on all nodes in the RAC configuration.

To set up the Oracle auto-login wallet

You can configure your Oracle database to use an auto-login wallet. See for more information about configuring the auto-login wallet on an Oracle RAC configuration. You must complete the auto-login wallet configuration on all nodes in the RAC configuration.

Integrating Oracle Data Guard Physical Standby with HSM on Demand Service

With Oracle Data Guard Physical Standby encrypted application data remains encrypted while redo log files are transferred between the primary and standby databases.

We recommend copying the primary wallet over to the standby site, so that in the event of a failover all data will remain available. The Oracle wallet can be configured to open automatically, making the master key available on the standby database as soon as it comes online.

This document provides detailed procedures on configuring a standby database for use with a primary database with TDE enabled using a HSM on Demand Service. There are two methods of approach:

> "Migrating an existing master key from the software wallet to HSM on Demand Service" below
> "Generating the master key directly on the HSM on Demand Service" on page 30

Migrating an existing master key from the software wallet to HSM on Demand Service

You can use an existing master encryption key in your Oracle Data Guard Physical Standby configuration. You can store the existing master encryption key in the HSM on Demand Service.

Steps to follow on primary database

1. Open the **sqlnet.ora** file in a text editor and update the following:

   ```
   ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY =
   /etc/oracle/wallet)))
   ```

2. Connect to the database as system.

   ```
   SQL> connect system/<password>
   ```

3. Migrate the wallet onto the HSM device.

   ```
   SQL> ADMINISTER KEY MANAGEMENT SET ENCRYPTION KEY IDENTIFIED BY "<HSM_partition_password>"
   MIGRATE USING <software_keystore_password> WITH BACKUP USING <backup_domain>;
   SQL> alter system set encryption key identified by <HSM_partition_password> migrate using
   <wallet_password>;
   ```

4. Create HSM auto-login wallet.

   a. Open the **sqlnet.ora** file in a text editor and update the following:

      ```
      ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA = (DIRECTORY =
      /etc/oracle/wallet)))
      ```

   b. Connect to the database as system.

      ```
      SQL> connect system/<password>
      ```

   c. Open the software keystore.

      ```
      SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY <software_keystore_password>;
      ```

   d. Add the HSM secret to the client.

```
SQL> ADMINISTER KEY MANAGEMENT ADD SECRET <HSM_partition_password> FOR CLIENT 'HSM_PASSWORD'
IDENTIFIED BY <software_keystore_password> WITH BACKUP USING <backup_domain>;
```

e. Close the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY <software_keystore_
password>;
```

f. Create the auto-login keystore.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE <keystore_location>
IDENTIFIED BY <software_keystore_password>;
```

g. Open the **sqlnet.ora** file in a text editor and update the following:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY =
/etc/oracle/wallet)))
```

h. Restart the database and connect as system.

5. Create HSM auto-login wallet

a. Navigate to the directory containing the software wallet.

b. Create an auto-login wallet.

```
# orapki wallet create -wallet . -auto_login
```

c. Add the following entry to the wallets to enable auto-opening of the HSM.

```
# mkstore -wrl . -createEntry ORACLE.TDE.HSM.AUTOLOGIN <any_non_empty_string>
```

d. Move the encryption wallet out of the ENCRYPTION_WALLET_LOCATION defined in the **sqlnet.ora** file.

e. Close the connection to the HSM.

```
SQL> alter system set encryption wallet close identified by <HSM_partition_password>
```

f. Open the connection to the HSM.

```
SQL> alter system set encryption wallet open identified by <HSM_partition_password>
```

NOTE This command will insert the <HSM_partition_password> into the auto-login wallet. As a result, if the wallet is open, no password is required to access data encrypted with the TDE master encryption key.

Steps to follow on standby database

1. Create a copy of the wallet directory on the primary database and copy its contents to the standby database.

```
# scp -r oracle@<hostname_or_IP_of_primary>:/etc/oracle/wallet/* /etc/oracle/wallet
```

2. Copy the **sqlnet.ora** file from the primary database to the standby database at $ORACLE_HOME/network/admin/.

```
# scp oracle@<hostname_or_IP_of_primary>:$ORACLE_HOME/network/admin/sqlnet.ora $ORACLE_
HOME/network/admin/sqlnet.ora
```

3. Start the database.

```
$ sqlplus / as sysdba
```

4. To resume managed recovery execute the following commands:

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION;
```

5. To switch the standby database into-read only mode execute the following commands:

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
SQL> ALTER DATABASE OPEN READ ONLY;
```

Generating the master key directly on the HSM on Demand Service

You can generate a master encryption key for your Oracle Data Guard Physical Standby configuration directly on the HSM on Demand Service.

Steps to follow on primary database

1. Close the keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY <HSM_partition_password>;
```

2. Open the **sqlnet.ora** file in a text editor and update the following:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY = <Path to
oracle wallet>)))
```

3. Create the software keystore in the location defined by the **sqlnet.ora** file.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/etc/oracle/wallet' IDENTIFIED BY <software_
keystore_password>;
```

4. Open the software keystore

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY <software_keystore_password>;
```

5. Add the HSM secret to the client.

```
SQL> ADMINISTER KEY MANAGEMENT ADD SECRET <HSM_partition_password> FOR CLIENT 'HSM_PASSWORD'
IDENTIFIED BY <software_keystore_password> WITH BACKUP USING <backup_domain>;
```

6. Close the software keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY <software_keystore_password>;
```

7. Create auto-login keystore.

```
SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE '/etc/oracle/wallet'
IDENTIFIED BY <software_keystore_password>;
```

8. Open the **sqlnet.ora** file in a text editor and update the following:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM) (METHOD_DATA = (DIRECTORY =
/etc/oracle/wallet)))
```

9. Restart the database and connect as system.

Steps to follow on standby database

1. Create a copy of the wallet directory on the primary database and copy its contents to the standby database.

```
# scp –r oracle@<hostname_or_IP_of_primary>:/etc/oracle/wallet/* /etc/oracle/wallet
```

2. Copy the **sqlnet.ora** file from the primary database to the standby database at $ORACLE_HOME/network/admin/.

```
# scp oracle@<hostname_or_IP_of_primary>:$ORACLE_HOME/network/admin/sqlnet.ora $ORACLE_HOME/network/admin/sqlnet.ora
```

3. Connect to the database as the administrator.

```
$ sqlplus / as sysdba
```

4. To resume managed recovery execute the following commands:

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION;
```

5. To switch the standby database into-read only mode execute the following commands:

```
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP MOUNT;
SQL> ALTER DATABASE OPEN READ ONLY;
```

Working with Pluggable Databases (Advisory Content)

A new feature for Oracle Database 12c is Multitenant Architecture, Oracle Multitenant delivers a new architecture that allows a multitenant container database to hold many pluggable databases. The multitenant architecture enables an Oracle database to function as a multitenant container database (CDB) that includes zero, one, or many customer-created pluggable databases (PDBs). A PDB is a portable collection of schemas, schema objects, and non-schema objects that appears to an Oracle Net client as a non-CDB. All Oracle databases before Oracle Database 12c were non-CDBs.

This Pluggable Databases reference section contains the following topics:

About Containers in a CDB

A container is either a PDB or the root container (also called the root). The root is a collection of schemas, schema objects, and non-schema objects to which all PDBs belong. Every CDB has the following containers:

> Exactly one root:
  - The root stores Oracle-supplied metadata and common users. A common user is a database user known in every container. The root container is named CDB$ROOT.
> Exactly one seed PDB:
  - The seed PDB is a system-supplied template that the CDB can use to create new PDBs. The seed PDB is named PDB$SEED. You cannot add or modify objects in PDB$SEED.
> Zero or more user-created PDBs:
  - A PDB is a user-created entity that contains the data and code required for a specific set of features. For example, a PDB can support a specific application, such as a human resources or sales application. No PDBs exist at creation of the CDB. You add PDBs based on your business requirements.

You can use PDBs to achieve the following goals:

> Store data specific to a particular application. For example, a sales application can have its own dedicated PDB, and a human resources application can have its own dedicated PDB.
> Move data into a different CDB
  - A database is "pluggable" because you can package it as a self-contained unit, and then move it into another CDB.
> Isolate grants within PDBs
  - A local or common user with appropriate privileges can grant EXECUTE privileges on a package to PUBLIC within an individual PDB.

We recommend using the Oracle DBCA for configuring your PDB.

Enabling TDE in PDBs

The following procedures detail the steps to take when configuring TDE to function on multiple PDBs.

To enable TDE in a PDB

1. Open the **tnsnames.ora** file in a text editor and add a new service for the PDB. Ensure that you configure the TNS_ADMIN environment variable to point to the correct **tnsnames.ora** file.

```
<pluggable_database_name> =
(DESCRIPTION =
   (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = <pluggable_database_name>.localdomain)
   )
)
```

2. Restart the listener service.

```
# lsnrctl stop
# lsnrctl start
```

3. Start a sqlplus session.

```
$sqlplus / as sysdba
```

4. Alter the PDB permissions.

```
SQL> ALTER PLUGGABLE DATABASE <pluggable_database_name> OPEN READ WRITE;
```

5. Connect to the PDB.

```
SQL> CONNECT system/<system_password>@<pluggable_database_name>;
```

6. Grant the following permissions to the PDB admin.

```
SQL> GRANT ADMINISTER KEY MANAGEMENT TO <PDB_admin>;
SQL> GRANT CREATE SESSION TO <PDB_admin>;
SQL> GRANT CONNECT TO <PDB_admin>;
SQL> GRANT DBA TO <PDB_admin>;
SQL> GRANT CREATE ANY TABLE TO <PDB_admin>;
SQL> GRANT UNLIMITED TABLESPACE TO <PDB_admin>;
```

7. Make the <PDB_admin> the default profile.

```
SQL> ALTER USER <PDB_admin> PROFILE DEFAULT;
SQL> COMMIT;
```

8. You are now able to connect to the PDB using the PDB user.

```
SQL> CONNECT <PDB_admin>/<pdb_admin_password>@<pluggable_databse_name>
```

Generating TDE Master Encryption Key for PDB

The following procedure set demonstrates using the HSM on Demand Service to generate a master encryption key for the PDB.

To generate a TDE master encryption key for PDB

1. Open the **sqlnet.ora** file in a text editor and update the following:

```
ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = HSM))
```

2. Start a sqlplus session.

```
sqlplus / as sydba
```

3. Connect to the PDB.

```
SQL> connect <PDB_admin>/<pdb_admin_password>@<pluggable_databse_name>
```

4. Run the ADMINISTER KEY MANAGEMENT SQL

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY <HSM_partition_password>;
```

5. Generate the PDB master encryption key.

```
SQL> ADMINISTER KEY MANAGEMENT SET IDENTIFIED BY <HSM_partition_password>
```

NOTE This command will generate a new master key. Any encryption or decryption operations performed within the pluggable database will use the pluggable database master key.

Using Oracle Wallet Manager (OWM) (Advisory Content)

You can use Oracle Wallet Manager for changing wallet passwords and enabling auto-login.

To use OWM to change the wallet password and enable auto-login

1. Start Oracle Wallet Manager.
2. Open the software-based wallet and click **Change Password**.
3. Enter the same password string that was used for the HSM wallet as the new password for the software-based wallet. Click **Save**, and **Exit**.
4. Connect to the database as system.

   ```
   $ sqlplus / as sysdba
   ```
5. Open the HSM and software wallet.

   ```
   SQL> alter system set encryption wallet open identified by "<HSM_partition_password>";
   ```
6. Close the wallet.

   ```
   SQL> alter system set encryption wallet close identified by "<wallet_password>";
   ```
7. Start Oracle Wallet Manager.
8. Open the software-based wallet, change the password back to the initial password. Enable the **Auto-Login** check box. Click **Save**, and **Exit**.
9. Verify that an auto-open wallet has been created in the oracle wallet directory. Rename the encryption wallet so that TDE will use the auto-login wallet.
10. Connect to the database as system.

    ```
    $ sqlplus / as sysdba
    ```
11. Open the HSM wallet.

    ```
    SQL> alter system set wallet open identified by "<HSM_partition_password>";
    ```